Numerical Algorithms
https://doi.org/10.1007/s11075-022-01479-5

ORIGINAL PAPER

Structured low-rank approximation for nonlinear matrices

Antonio Fazzi[1,2]

Received: 27 December 2021 / Accepted: 1 December 2022

Abstract
Structured low-rank approximation problems are very popular in the scientific community since they are involved in several applications in different scientific fields. The search for new algorithms which improve the performance of the existing ones is still an active research topic. However, almost all the existing results focus on the class of linearly structured matrices. Motivated by two applications in the field of computer algebra and system identification, respectively, we propose an algorithm for structured low-rank approximation of nonlinearly structured matrices. The idea comes from an extension of the same method studied and tested by the author in the linear case.

Keywords Structured low-rank approximation · Nonlinearly structured matrices · Structured matrix perturbation · Approximate common factors computation

Mathematics Subject Classification (2010) 41A29 · 65K10 · 65K05

1 Introduction

1.1 Preliminaries

The approximation of a matrix with another one of lower rank is a well known and studied topic in numerical linear algebra, computer vision, and other fields of science because of the applications in mathematical modeling, signal processing, data compression, etc. The problem consists in the minimization of an objective functional which measures the error between the data matrix (whose entries are usually affected by noise) and the approximating one, which is subject to a rank constraint. The clas-

✉ Antonio Fazzi
antonio.fazzi@unipd.it

[1] Department ELEC, Vrije Universiteit Brussel, Pleinlaan 2, Brussels, 1050, Belgium

[2] Department of Information Engineering, University of Padova, via Gradenigo 6/b, Padova, 35131, Italy

Published online: 15 December 2022

sical formulation of the problem is as follows: given a matrix D of dimension $m \times n$, and an integer $r < \min(m, n)$, compute

$$\min_{\hat{D}} \|D - \hat{D}\|_F \quad \text{subject to} \quad \text{rank}(\hat{D}) \leq r, \tag{1}$$

where a common choice for measuring the distance is the Frobenius norm. The solution of (1) can be computed analytically, and the result is known in the literature as Eckart-Young theorem [1].

But several applications need the data matrix, as well as the approximating one, to be structured, that is their entries have to respect a given pattern; hence, the rank constraint only on the output matrix is not enough to compute an admissible solution for the considered problem. The structure specification is a function which maps a vector of length n_p into the set of $m \times n$ matrices. By adding the constraint on the structure of the matrix, the structured low-rank approximation problem reads as follows:

Problem 1 *Given a structure specification $S : \mathbb{R}^{n_p} \to \mathbb{R}^{m \times n}$, a vector $p \in \mathbb{R}^{n_p}$ such that $S(p)$ is full rank, and a natural number $r < \min(m, n)$, compute*

$$\min_{\hat{p}} \|p - \hat{p}\|_2 \quad \textit{subject to} \quad \textit{rank}(S(\hat{p})) \leq r \tag{SLRA}$$

These problems are usually non-convex optimization problems for which there is no analytical solution, neither a standard solution method. The study and development of algorithms for structured low-rank approximation problems are still an active research topic and it has been widely studied in the case of linear structure specifications [2, 3] like Toeplitz and Hankel, or more generally for matrices having an affine structure [4, 5]. But the case of nonlinear matrices is missing from the literature because the presence of nonlinearities in the data does not allow to naturally adapt the same approaches from the linear case. Only some particular problems (curves and hypersurfaces fitting [6–9]) have been considered and few algorithms were extended from the case of linear structures [10, 11]. A novel algorithm based on the solution of deconvolution equations (not optimization based) has recently been proposed for the problem of Vandermonde low-rank approximation [12].

The paper considers Problem 1 for nonlinear structure specifications. But the proposed algorithm is inspired by an iterative methodology used for the numerical solution of several structured matrix nearness problems with linear structure specifications: approximate common factors computation of scalar and matrix polynomials [13–15] and Hankel low-rank approximation [16].

1.2 The gradient system methodology for linearly structured low-rank approximation

We briefly describe here the main points of the numerical methodology proposed for the solution of some structured low-rank approximation problems involving linearly structured matrices: Sylvester matrices [13–15] and Hankel matrices [16]. This is helpful to better understand the whole numerical scheme we are going to propose and

to highlight the issues arising in the passage from the linear to the nonlinear case. We denote as S_L a Sylvester or a Hankel matrix.

The problem to be solved is still Problem (SLRA), where the structure specification is S_L. The perturbed matrix $S_L(\hat{p})$ is of the form

$$S_L(\hat{p}) = S_L(p) + \epsilon S_L(\Delta p), \tag{2}$$

where

- ϵ is a scalar which measures the norm of the perturbation;
- $S_L(\Delta p)$ is a normalized perturbation matrix having the same structure of the starting matrix.

The splitting of the perturbation allows to update the two factors independently on two iteration levels. At the inner level, the norm of the perturbation is fixed and we aim at minimizing the kth singular value of $S_L(p) + \epsilon S_L(\Delta p)$, where k is the sought rank reduction. This is done by looking for a stationary point of the gradient system

$$\dot{E} = -P_{S_L}(wv^T) + \langle P_{S_L}(wv^T), E\rangle E \quad E = S_L(\Delta p), \tag{3}$$

where P_{S_L} is the operator which orthogonally project the argument on the structure S_L (it can be computed explicitly by averaging the diagonals/antidiagonals of the argument matrix) and w, v are the singular vectors associated with the singular value to be minimized. We remark that (3) describes the steepest descent direction for σ_k over the ball of matrices having (Frobenius) norm at most ϵ. Observe that, since the structure is linear, all the matrices in (3) have the same structure; hence, we can state (3) directly on the vectors which generate the structured matrices.

At the outer level, once we computed the minimizer at the previous step, we need to compute the smallest value ϵ^*, that is the minimum norm of an admissible solution which achieves the sought rank constraint

$$\epsilon^* = \min\{\epsilon \mid \sigma_k(S_L(p) + \epsilon S_L(\Delta p)) = 0\}, \tag{4}$$

where $S_L(\Delta p)$ is the minimizer computed by integrating (3). The computation in (4) is performed by using a suitable root-finding algorithm.

The two iteration levels are repeated until a predefined stopping criterion based on a fixed small tolerance is satisfied. Such a two-steps methodology performs quite well in comparison with other existing algorithms, especially in terms of accuracy on the computed solutions. This motivated the extension to nonlinear structures, which is the topic presented in the paper.

1.3 Content of the paper

The problem we aim at solving is Problem 1 in the case when the structure specification S is nonlinear. The key idea of the proposed algorithm is to perturb the vector p until the perturbed structured data matrix satisfies the sought rank constraint by preserving the (nonlinear) structure at the same time. This is done by moving the singular values of the data matrix in a structured way, by following a descent direction, till the problem constraints are satisfied.

However, we cannot use directly the methodology summarized in Section 1.2 since some issues arise in the passage from linear to nonlinear structure specifications due to the loss of some important properties for the considered set of matrices; the key idea of minimizing a certain functional associated with the singular values of a structured matrix is carried over, though.

We underline that the presence of nonlinear functions in the data is a novelty in the framework of structured low-rank approximation: this allows to extend the classical formulation to a wider class of problems and applications. The weak point (of this current version) of the algorithm, however, is the fact that some Matlab built-in optimization functions play a key role in the structure preserving property.

The organization of the paper follows: In Section 2, we show two applications which motivated this work: Vandermonde low-rank approximation (Section 2.1) and the identification of systems generated by nonlinear maps (Section 2.2). In Section 3, we propose the algorithm for solving Problem 1, pointing out the differences with respect to the linear case summarized in Section 1.2. In Section 4, we show some numerical simulations for the applications illustrated in Section 2, and finally we sum up the results and we propose some possible directions for future research in Section 5.

2 Applications

First of all, we show few applications which motivated the development of the work. The first is the structured low-rank approximation of Vandermonde matrices in the framework of the popular *Approximate polynomials common factor computation* problem. The second is in the area of system identification, and it is the recovery of a time series (generated by a nonlinear map) from its noisy version.

2.1 Vandermonde low-rank approximation

Vandermonde matrices are usually associated with polynomials evaluation and interpolation problems and Vandermonde systems of equations arise in applications in exponential data modeling [10, 17]. The motivating application we are interested in comes from computer algebra, and it is the computation of (approximate) common factors of polynomials. This problem was already presented in [12] as a Vandermode low-rank approximation problem and it was solved by a numerical method based on the solution of some deconvolution equations. We recall the problem here, and we plan to solve it via the optimization-based algorithm we are going to propose.

A square Vandermonde matrix has the form

$$V = V(c) = \begin{pmatrix} 1 & \cdots & 1 \\ c_1 & \cdots & c_n \\ c_1^2 & \cdots & c_n^2 \\ \vdots & \cdots & \vdots \\ c_1^{n-1} & \cdots & c_n^{n-1} \end{pmatrix} \tag{5}$$

where the (i, j)th entry of the matrix is the $(i-1)$th power of the jth entry of the parameter vector c. It is well known that a Vandermonde matrix is rank-deficient if and only if (at least) two of the generating parameters (the entries of c) coalesce. These coalescing parameters can detect the presence of common factors between two (or more) polynomials. In particular, let $r_1 \in \mathbb{R}^n, r_2 \in \mathbb{R}^m$ be the set of roots of two univariate polynomials $p_1(x)$, $p_2(x)$, respectively, and collect them in one vector $c = [r_1, r_2]$. Assuming that the polynomials have simple roots, we can say that p_1 and p_2 are coprime if and only if the entries of r_1 and r_2 are pairwise different, that is, all the entries of c are different.

A popular problem in computer algebra is the computation of the distance to common divisibility for a pair of coprime polynomials, which arises in several applications in system and control theory [18–20]. The problem is known in the literature with different names (e.g., approximate greatest common divisor computation or approximate common factors computation) depending on the slightly different formulations.

The novelty already introduced in [12] and remarked here is the definition of distance of polynomials to common divisibility. The popular definition of such a distance considered in the literature is based on the polynomials coefficients (we cite as examples [13, 14, 21–26]). But the coefficients of a polynomial, differently from its roots, are not uniquely defined: indeed, the multiplication of a polynomial by a nonzero constant changes the coefficients but not the roots. The use of Vandermonde matrices allows to restate the distance to common divisibility in terms of distance between the polynomials roots instead of the coefficients. Looking back at our problem, it consists in computing a vector $\tilde{c} = [\tilde{r}_1, \tilde{r}_2]$, as close as possible to c, such that $\tilde{r}_{1,i} = \tilde{r}_{2,j}$ for (at least) a pair i, j. This makes the Vandermonde matrix $V(\tilde{c})$ rank-deficient.

2.2 Identification of a time-series generated by a nonlinear map

We show here another interesting application in the framework of system identification. Identification problems are quite popular in the structured low-rank approximation framework, and in particular a well-known result states that the identification of a linear time-invariant system is equivalent to the low-rank approximation of a Hankel matrix built from an observed trajectory [27]. However, the Hankel structure restricts this result to the class of linear systems, and only few methods exist in the literature to extend this approach to some special classes of nonlinear systems [28–30]. The idea is to extend the structured low-rank approximation framework to the identification of systems generated by nonlinear maps.

2.2.1 Singular value decomposition and nonlinear matrices

Singular value decomposition is a powerful linear algebra tool which has applications in data analysis, statistics, signal processing, and other different fields. The usefulness of this matrix decomposition in the following is twofold: on the one hand, it reveals the rank of the involved (nonlinear) structured matrix, and on the other, it helps to understand which is the relation among the entries of the involved matrix (the

image of the function S in Problem 1) and the meaning of the rank constraint, that is the motivation of the problem. After a brief reminder on the singular value decomposition, we show how such a decomposition detects the relation among the entries of a certain (structured) nonlinear matrix. We follow the arguments described in [31].

The singular value decomposition (SVD) is the factorization of a matrix $A \in \mathbb{C}^{m\times n}$ into three matrices: a unitary matrix $U \in \mathbb{C}^{m\times m}$, a real diagonal matrix $\Sigma \in \mathbb{R}^{m\times n}$, and a unitary matrix $V \in \mathbb{C}^{n\times n}$ [32]. We can write

$$A = U\Sigma V^*,$$

where V^* denotes the complex conjugate of the matrix V. One of the very important features of this factorization is the fact that it allows generalizing several definitions and computational techniques from square to rectangular matrices.

Consider a matrix A of the form

$$A = [L, f_1(L), \ldots, f_i(L)], \tag{6}$$

where L is a linearly structured matrix and $f_1, \ldots, f_i$ are some arbitrary (differentiable) nonlinear functions of the columns of L (polynomial, sinusoidal, exponential, etc.). The basic idea lies on the fact that a nonlinear relationship among the entries of A can be interpreted as a linear relationship among these nonlinear columns. The tool to find such a link is the svd factorization $A = U\Sigma V^*$. If the smallest singular value of A is null, by expanding with respect to the last column, we get

$$\sum_p A_j^p V_p^i = 0 \quad \forall j.$$

that is, the columns of A span a linear vector space; this allows to recover the dependence between the linear and the nonlinear columns of the matrix A.

2.2.2 Structured matrices in system identification

One application of singular value decomposition is in the field of system identification: we want to detect and estimate the parameters of a time series $\{X\}$ which is the trajectory generated by a certain system. If the generating system is linear and time-invariant of order k, a well-known result states that the Hankel matrix $H_{k+1}(X)$ built from the data trajectory and having $k+1$ rows is rank-deficient [27]. A similar result holds true in the case of nonlinear systems by embedding the Hankel matrix into a bigger (structured) matrix having additional columns associated with the nonlinear behavior of the system.

Consider a time series $X = [X_1, \ldots, X_T]$ which is generated by

$$X_{n+1} = g(X_n, X_{n-1}, \ldots) \tag{7}$$

for a nonlinear function g (g can be a combination of polynomial, sinusoidal, or exponential functions). Following (6), we need to build an appropriate structured matrix which can detect the original time series X generated by g. The linear part is a Hankel matrix generated from X, while the nonlinear columns $f_1, \ldots, f_i$ are a set of

basis functions generating all the nonlinearities in g (this assumes some preliminary knowledge on the function g, though):

$$S = (H(X),\ f_1(X),\ f_2(X), \ldots,\ f_i(X)). \tag{8}$$

If X is generated as in (7), the structured matrix in (8) is rank-deficient. If X is noisy, the matrix S in (8) is full rank; the noiseless signal X can be recovered by solving a nonlinear structured low-rank approximation problem.

3 The algorithm

The proposed algorithm to solve Problem 1 is based on a double iteration method similar to [13, 16], but the considered structure specification is now nonlinear, so we need to change and adapt some steps of the algorithm. To fix the ideas, we consider a rank reduction by one, but the same idea can be naturally extended to deal with greater rank reductions. The problem is stated directly on the parameter vector (the vector generating the structured matrix) as in (SLRA); therefore, the perturbed structured matrix will have the form $S(p+\epsilon e)$, where ϵ is a scalar measuring the distance between the computed and the starting vectors, while e is a (normalized) perturbation on the data vector which (locally) minimizes the smallest singular value over the ball of vectors of fixed norm ϵ. The vector e and the scalar ϵ are updated independently on two different levels: we explain in details in the following how to compute both the perturbations.

3.1 Descent direction for the smallest singular value

The considered problem is the approximation of a given structured matrix (whose structure is nonlinear in the generating parameters) with another matrix, as close as possible to the starting one (according to the given norm), which satisfies both a structure preserving property and a constraint on the rank. The mathematical formulation of the problem is written in Problem 1 with S nonlinear. In this section, we describe the inner level minimization of the proposed methodology, that is we fix the value of the norm of the perturbation ϵ, and we aim at minimizing the smallest singular value of the structured matrix $S(\tilde{p}) = S(p+\epsilon e)$ over the vectors ϵe whose norm is ϵ. We remark that the norm of the perturbation ϵ on the parameter vector is linked to the norm of the structured matrix and such a relation depends on the definition of distance, that is the given norm. For convenience, we call $\bar{\epsilon}$ the norm of the distance $\|S(p+\epsilon e) - S(p)\|_F$.

The presence of a nonlinear structure specification S makes the minimization problem more difficult compared to other (linear) structured low-rank approximation problems solved by an analogous gradient system approach [13, 15, 16] (see also Section 1.2); we list in the following some important properties which are not preserved in the passage from linear to nonlinear structure specifications, and therefore the corresponding points in the algorithm need to be modified in order to accomplish the needed task. As a consequence, we are not able to write explicitly a gradient system (as in (3)) whose stationary points give the sought minimizer.

1. Differently from what happens in (3), where E preserves the same structure as the data matrix, in the nonlinear case, the perturbation matrix $E = S(p + \epsilon e) - S(p)$ does not have the same structure as $\mathcal{S}(p)$, that is E is not in the range of $\mathcal{S}$ (the set of nonlinear structured matrices is not closed under the sum, $S(p+q) \neq S(p) + S(q)$);
2. it can be difficult to work directly with the structured matrix $S(p) + \bar{\epsilon}E$ at the inner level iteration: the perturbation matrix E pointing towards a descent direction for the considered singular value to be optimized cannot be normalized simply by dividing its entries by the same scalar. This is because of the nonlinear relation among some of its entries (this would not preserve the nonlinear structure);
3. we are not able to find an explicit formula for computing the operator P_S (which finds the closest nonlinear structured matrix to an arbitrary given matrix) in the nonlinear case. On the other hand, the operator P_{S_L} in (3) could be explicitly computed for Toeplitz and Hankel matrices by diagonal or antidiagonal averaging (see [13, 16]).

The previous points summarize the main differences and issues arising in the case of nonlinear structure specifications. In particular, the first two points highlight the reason why we prefer to work directly with the generating parameter vector (the structured matrix is needed only to compute its singular value decomposition). The last point is solved by built-in Matlab optimization functions. It can happen that an explicit formula for the perturbation vector exists, but only for some special structure specifications, hence it would not be general. Once we understood which are the main points of the methodology to be adapted in the nonlinear case, we can go on with the details of the inner level minimization, that is the minimization of the smallest structured singular value over a fixed norm level set.

A key result of this numerical methodology is a classical theory about eigenvalues perturbation of a matrix, which is then extended to the case of singular values. Despite the structure specification is nonlinear, it has been shown in [33] that an analogous result from the linear case still holds true for nonlinear matrices.

Lemma 1 *Consider the differentiable matrix valued function $C(t)$ for t in a neighborhood of 0. Let $\lambda(t)$ be an eigenvalue of $C(t)$ converging to a simple eigenvalue λ_0 of $C_0 = C(0)$ as $t \to 0$. Let y_0 and x_0 be left and right eigenvectors, respectively, of C_0 corresponding to λ_0, that is, $(C_0 - \lambda_0 I)x_0 = 0$ and $y_0^*(C_0 - \lambda_0 I) = 0$. Then, $y_0^* x_0 \neq 0$ and $\lambda(t)$ is differentiable near $t = 0$ with*

$$\dot{\lambda}(0) = \frac{y_0^* \dot{C}(0) x_0}{y_0^* x_0}. \tag{9}$$

Without loss of generality we will always assume $\|x_0\|_2 = 1$, $\|y_0\|_2 = 1$, $y_0^* x_0 > 0$.

Lemma 1 is stated in matrix form, so we need to apply this result to a (structured) perturbed matrix of the form $S(\tilde{p}) = S(p) + \bar{\epsilon}E$. We remark again that the perturbation matrix E cannot be written as a matrix in the range of S, because the sum of two matrices in the range of S would not have the same structure

($S(p)+S(q) \neq S(p+q)$), but E is a structured matrix too (whose structure is different from S). The idea is to move then the perturbation on the parameter vector p to achieve our goal; the structure specification S is needed only to compute the singular value decomposition.

The first step is to state a result analogous to Lemma 1 for singular values. If we apply Lemma 1 to a positive semidefinite matrix of the form $S(\tilde{p})^T S(\tilde{p})$, where $S(\tilde{p}) = S(p) + \bar{\epsilon} E$ (we omit the time dependence in the following), some computations lead to a similar expression for the derivative of a singular value of the matrix $S(\tilde{p})$. If σ is a differentiable singular value of the matrix function $S(\tilde{p})$ (the differentiability can be guaranteed by assuming that there are no coalescing singular values) and u, v are the corresponding left and right singular vector, respectively, then

$$\begin{aligned} \frac{d}{dt}\sigma^2 &= v^T \frac{d}{dt}(S(\tilde{p})^T S(\tilde{p}))v = 2\sigma u^T \dot{E} v \\ \frac{d}{dt}\sigma &= u^T \dot{E} v. \end{aligned} \tag{10}$$

Since the rank-deficiency of a matrix is equivalent to a zero singular value, we are interested in minimizing (up to a fixed small tolerance) the smallest singular value of $S(\tilde{p})$. The minimization problem is solved by looking at the zeros of the derivative of σ (10), in order to find a descent direction and a candidate local minimum.

By rewriting the expression of the derivative, such a minimum is achieved by the minimum of the function

$$u^T \dot{E} v = \langle u v^T, \dot{E} \rangle, \tag{11}$$

where $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product.

Theorem 1 *The unstructured gradient for the smallest singular value of $S(\tilde{p})$ is given by the rank one matrix $\dot{E} = -uv^T$.*

Proof By construction (11) is the derivative of the smallest singular value of the perturbed matrix $S(\tilde{p}) = S + \bar{\epsilon} E$, and $\dot{E}$ is the (unknown) minimizer. Since we deal with the Frobenius metric, an inner product is minimum when the two arguments are the same in modulus but they have opposite signs. □

Theorem 1 gives the expression for the free (unconstrained) gradient associated with the smallest singular value; however, the addition of such a rank one matrix to the data matrix does not preserve, in general, the structure. As anticipated, we are not aware of an explicit formula which express the closest nonlinear structured matrix to the rank one matrix uv^T. Hence, we need to compute numerically a perturbed vector $\tilde{p} = p + \epsilon e$ such that the structured matrix $S(\tilde{p}) = S(p + \epsilon e)$ is as close as possible to the (unstructured) matrix $S(p) - \bar{\epsilon} u v^T$. The structured minimization problem is approached by the use of built-in Matlab optimization functions. In particular, the (local) optimization problem is solved by computing a perturbation which accomplish for the solution of the minimization problem directly on the parameter vector p. We describe in the following paragraph the details of this constrained structured perturbation.

Structured perturbation via Matlab optimization Given a structured matrix $S(p)$, an unstructured perturbation E (for the considered problem $E = -\bar{\epsilon} u v^T$), and a scalar ϵ, we show here a computational method (based on the Matlab optimization function *fmincon*) to get a structured matrix $\mathcal{S}(p+q)$ which minimizes the smallest singular value over the ball of vectors q whose norm is bounded by ϵ and it is as close as possible to $S(p) + E$.

The Matlab function *fmincon* is a built-in function for solving constrained nonlinear optimization problems. The inputs we need to provide to *fmincon* are the objective function to be minimized, the constraint (the norm of the perturbation vector), and a starting point. The output is the direction (the derivative of the perturbation vector e) which makes the singular value σ decreasing. We summarize the pseudo-code in Algorithm 1 and we discuss the main steps below.

Require: S structure specification, p starting vector, e current perturbation, ϵ norm of the computed perturbation, u, v singular vectors of $S(\tilde{p})$ associated with σ
Ensure: $\dot{e}$ direction for the perturbation vector
1: Find the structure of $\dot{E}$
2: Problem: $\min_x u' \dot{E}(x, p, e) v$
Constraint: $\|x\|_2 = \epsilon$
Starting point: $x_0 = 0$
3: Solution:
$\dot{e}$ is the solution of the given constrained problem using the function *fmincon* starting from x_0

Algorithm 1 Structured perturbation via *fmincon*.

The function we need to minimize is given in (11), but we need an adjustment since we want to optimize on the parameter vector and not on the structured matrix. This is done by observing that $\bar{\epsilon} E = S(\tilde{p}) - S(p)$ so, starting from S, we can explicitly compute also the structure of the perturbation matrix $E = E(p, e)$ and its derivative $\dot{E}(p, e, \dot{e})$ (this method works fine and is efficient whenever the exact expressions for the derivatives of the entries of E can be easily computed and the dimensions of the involved matrices are not too big, otherwise a different strategy should be investigated). We also need the vectors u, v which come from the svd of the matrix $S(\tilde{p})$. The problem constraint is simple and it consists simply on binding the norm of the solution vector. As initial point we simply choose the zero vector. This last choice could be not the best one but it guarantees to get the same numerical result each time we run the algorithm on the same data. Another possibility could be to choose several (random) initial points and take as solution the one corresponding to the minimum of the objective function; this strategy is more robust and it increases the probability of getting closer to the global minimizer, but it is computationally more expensive since we need to solve several times the same optimization problem starting from different initial values.

At this point, we know how to compute the sought structured matrix which makes the objective functional decreasing: we simply add (a multiple of) the direction computed by Algorithm 1 to the previous value of the perturbation vector. We observe that the explicit computation of the structures of the matrices E, $\dot{E}$ in the Matlab function *fmincon* replaces the use of the structure specification S. The distance problem in Algorithm 1 needs to be solved iteratively in the integration of a gradient system which monotonically decreases the considered singular value until it reaches a point of local minimum. However, as we already understood in the previous discussion, we are not able to write down explicitly an equation which describes such a gradient dynamic. This issue is considered in the following section.

3.2 Numerical integration

We want to integrate numerically a suitable gradient system which computes the (normalized) vector e such that the smallest singular value σ of the structured matrix $S(p + \epsilon e)$ decreases as much as possible. Since the value of ϵ is fixed, we can only update the vector e until we reach a point of (local) minimum. The computation is performed by adding to the current value for e the gradient direction computed by Algorithm 1. The sought minimum is a point where the value of σ does not decrease anymore; this is a zero of the derivative $\dot{\sigma}$, that is a zero of $\dot{E}$ or equivalently $\dot{e}$.

The whole integration of such a gradient system is performed, in this way, directly on the parameter vector p, but the stopping criterion needs to be stated on the structured matrix since it involves the computation of its singular values. The numerical scheme is shown in Algorithm 2.

We add few comments about Algorithm 2. The goal is to compute an optimal direction e which minimizes the value of σ over the ball of vectors whose norm is ϵ. We chose to use the Explicit Euler scheme because of the expensive method for the computation of the singular value σ (the svd), but different integration schemes can be used as well as different (cheaper) algorithms for the computation of the singular triplet, possibly exploiting the structure specification S. The addition of the step control strategy is needed in order to guarantee the monotonicity for the descent of σ.

Before going on with the outer level iteration (the update of ϵ), we summarize the computations performed at the inner level for a fixed value of ϵ. The goal is to compute an optimal normalized perturbation vector e which minimizes the smallest singular value of the matrix $S(p + \epsilon e)$; this is done iteratively by computing the gradient direction $\dot{e}$ in order to update the value of e until a convergence condition is satisfied. By using Algorithm 2, we compute a set of perturbation vectors $e_1, e_2, \ldots, e_{l-1}, e_l$ and the associated singular values $\sigma_1 > \sigma_2 > \ldots \sigma_{l-1} = \sigma_l$ ($\sigma_i = \sigma(S(p + \epsilon e_i))$).

3.3 The update of ϵ

At this point, we are able to compute a local extremizer e which corresponds to a (local) minimum of the singular value σ for the given value of ϵ. The next step is to

Require: p starting parameter vector, S (nonlinear) structure specification, ϵ norm of the perturbation, h step Euler method, γ step size reduction, tol stopping tolerance, e (optional) starting perturbation vector.
Ensure: $\tilde{p}$
1: **if** e is given in input **then**
   Set $\tilde{p} = p + \epsilon e$
2: **else**
   Set $\tilde{p} = p$
3: **end if**
4: Build the matrix $S(\tilde{p})$
5: Compute the singular triplet u, v, σ for the matrix $S(\tilde{p})$
6: Compute $\dot{e}$ by using Algorithm 1
7: Euler step $\rightarrow e = e + h\dot{e}$
8: Update $\tilde{p} = p + \epsilon e$
9: Compute the singular triplet u^t, v^t, σ^t for the matrix $S(\tilde{p})$
10: **if** $\sigma^t > \sigma$ **then**
11: reject the result and reduce the step h by a factor γ
12: go to line 7
13: **else**
14: accept the result
15: **if** $\sigma^t < tol$ or $\|\sigma - \sigma^t\| < tol$ **then**
16: **return**
17: **end if**
18: Set $u = u^t, v = v^t, \sigma = \sigma^t$
19: **end if**
20: Repeat from line 6

Algorithm 2 Numerical integration of the gradient system for σ over a fixed norm level set — Explicit Euler scheme.

update such a value of ϵ in order to get closer to an admissible solution for Problem 1.

We denote as $\sigma(\epsilon)$, $e(\epsilon)$ the extremizers (the outputs of Algorithm 2) corresponding to the given value ϵ, and we call ϵ^* the smallest (optimal) value of ϵ such that $\sigma(\epsilon^*) = 0$ or equivalently the corresponding matrix $S(p+\epsilon^* e(\epsilon^*))$ is rank-deficient. First of all, we need to choose a starting value ϵ_0. A good choice can be the smallest singular value of the starting matrix $S(p)$, that is the unstructured distance to singularity, since we expect that the structured distance to singularity is bigger than the unstructured one.

We discuss in the following different possible strategies to find the sought value ϵ^*, or at least a good approximation for it. The difficulty in the use of first order optimization methods to compute ϵ^* lies in the search for an explicit formula for the derivative of σ with respect to ϵ. This is the same issue we had at the inner iteration level, where we were not able to write a formula for the gradient system minimizing

the structured singular value. However, at this point, we look for a computationally cheap update of ϵ, so we prefer to avoid the use of external functions for this computation.

A simple strategy can be a bisection of a suitable interval $[\epsilon_0, \epsilon^u]$, where ϵ^u is an upper bound. However, the downside is that possible big jumps in the value of σ may return misleading results because the presence of nonlinearities and the possible ill-conditioning of the involved matrices (e.g., Vandermonde matrices, see Section 2.1) can totally change the values of the starting parameters. The adopted strategy is the simplest and it seems very efficient (at least for this version of the algorithm): we just start from the starting point ϵ_0 and we increase at each step the value of ϵ by a small constant which depends on the value $\sigma(\epsilon_{it-1})$, where the last notation is used to denote the value of σ computed at the previous iteration. By doing so, we get both the continuity and the monotonicity for the singular value σ with respect to the integration of the gradient system (for a fixed value of ϵ) and the iteration levels (the increasing values of ϵ). This is because small changes in the perturbation norm leads to small changes in the entries of the computed solutions. Moreover, the numerical simulations show that the computed value ϵ^* is small, so a bisection strategy would not be so efficient unless a very good estimate for the upper bound ϵ^u is available. We show in Algorithm 3 the main scheme of the whole algorithm.

Require: p starting parameter vector, S (nonlinear) structure specification, , tol stopping tolerance.
Ensure: $\tilde{p}$
1: Compute σ via the svd factorization of $S(p)$
2: Set $\epsilon = \sigma$
3: Run Algorithm 2 with data p, S, ϵ to minimize the smallest singular value σ of $S(p + \epsilon e)$
4: Store the output vector $\tilde{p} = p + \epsilon e$ and the singular triplet u, v, σ of the matrix $S(\tilde{p})$
5: **while** $\sigma > tol$ **do**
6: $\epsilon = \epsilon + \sigma$
7: Run Algorithm 2 with data $\tilde{p}, S, \epsilon$
8: Update $\tilde{p}$
9: Compute σ via the svd factorization of $S(\tilde{p})$
10: **end while**

Algorithm 3 Structured low-rank approximation of a nonlinear matrix.

Finally, in order to have more clear ideas on how Algorithm 3 works, we illustrate in the following what happens. For each fixed value of the norm of the perturbation ϵ, we compute a set of decreasing singular values until a stationary point is reached;

the norm of the perturbation ϵ is then increased to get closer to a zero singular value (i.e., an admissible solution for the considered problem).

$$\begin{aligned} \epsilon_0 &\to \sigma_{01}, \sigma_{02}, \ldots, \sigma_{0l_0} \\ \epsilon_1 &\to \sigma_{11}, \sigma_{12}, \ldots, \sigma_{1l_1} \\ &\vdots \\ \epsilon^* &\to \sigma_1^*, \sigma_2^*, \ldots, \sigma_{l_*}^* \end{aligned} \tag{12}$$

3.4 Justification for the double iteration strategy

The use of the Matlab minimization function *fmincon* in Algorithm 1 is unusual, but it plays an important role in the proposed methodology. An attentive reader may wonder why Algorithm 1 is iterated instead of solving Problem 1 by setting one Matlab minimization problem only on the starting data. This question is meaningful, but we need to remember that we want to approach Problem (SLRA), that is a nonconvex optimization problem, by a local optimization strategy; therefore, unless we provide a *good* initial approximation to the Matlab function, we cannot expect to find an accurate solution (i.e., an admissible solution which locally minimizes the cost function (SLRA)). We remark that structured low-rank approximation problems are *difficult* to solve already for linearly structured matrices.

The proposed methodology, on the other hand, by computing minimizers of increasing norm, achieves to compute solutions which do not show big changes with the increasing of the iteration number. There are two main reasons which support this fact

1. the minimizer e, computed at each (inner) iteration step, is a good approximation for the solution to be computed in the successive iteration;
2. the choice for the update of ϵ (see Section 3.3) makes it possible to have a *smooth* transition between two successive iterations.

4 Numerical examples

The goal of this section is to test the proposed algorithm on the applications presented in Section 2 involving nonlinear structured low-rank approximation: Vandermonde low-rank approximation (Section 2.1) and identification of time series generated by nonlinear maps (Section 2.2).

4.1 Vandermonde low-rank approximation

The problem is the one described in Section 2.1, that is the approximation of a full rank Vandermonde matrix with a rank deficient one by coalescing two generating parameters. The parameters generating the starting Vandermonde matrix come from

the roots of the following polynomials of degree 2 (expressed by the vectors of their coefficients):

$$\begin{aligned} p_1 &= (1,\ -3.2,\ 2.2) \quad r_1 = (1,\ 2.2) \\ p_2 &= (1,\ -6.5,\ 10) \quad r_2 = (2.5,\ 4). \end{aligned} \tag{13}$$

The starting Vandermonde matrix is

$$V = V([r_1, r_2]) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2.2 & 2.5 & 4 \\ 1 & 2.2^2 & 2.5^2 & 4^2 \\ 1 & 2.2^3 & 2.5^3 & 4^3 \end{pmatrix} \tag{14}$$

whose singular values are 69.208, 3.981, 0.681, and 0.023. In order to apply the algorithm described in Section 3, we need, first of all, to understand which is the structure of the matrix $\dot{E}$. We start from the computation of the entries of $V(p+e)$ (we can write this directly in vectorial form since all the columns have the same structure):

$$V(p+e) = \begin{pmatrix} 1 \\ p+e \\ (p+e)^2 \\ (p+e)^3 \end{pmatrix} = V(p) + \begin{pmatrix} 0 \\ e \\ 2pe + e^2 \\ e^3 + 3p^2e + 3pe^2 \end{pmatrix} = V(p) + E(p, e).$$

At this point, $\dot{E}(p, e, \dot{e})$ comes simply by the derivatives of the entries of $E(p, e)$ (remember that p is constant):

$$\dot{E} = \begin{pmatrix} 0 \\ \dot{e} \\ 2(p+e)\dot{e} \\ 3(e^2 + p^2 + 2pe)\dot{e} \end{pmatrix}. \tag{15}$$

By plugging the structure of $\dot{E}$ into Algorithm 1, we are able to run Algorithm 3; we get our numerical solution by iteratively decreasing the smallest singular value of the given Vandermonde matrix (14) up to a fixed small tolerance of 10^{-6} (see Fig. 1).

The closest roots of the two polynomials in (13) (the ones which are expected to coalesce once we run the algorithm) are 2.2 and 2.5.[1] The computed solution is a vector $\tilde{r}$ such that the Vandermonde matrix $V(\tilde{r})$ can be considered rank-deficient.

The entries of the output vector $\tilde{r}$, approximated to the third decimal digit, are (1.060, 2.353, 2.353, 3.982), so we found the expected approximate common root between the polynomials in (13). The perturbed polynomials associated with the computed roots are:

$$\begin{aligned} \tilde{p}_1 &= (1,\ -3.413,\ 2.494) \quad \tilde{r}_1 = (1.060, 2.353) \\ \tilde{p}_2 &= (1,\ -6.335,\ 9.370) \quad \tilde{r}_2 = (2.353, 3.982). \end{aligned} \tag{16}$$

and the distance between the starting and the computed roots is $\|[r_1, r_2] - [\tilde{r}_1, \tilde{r}_2]\|_2 = 0.221$.

To conclude the example, for the sake of completeness, we run a comparison with the algorithm presented in [13] which still solves an approximate common factor

[1] The algorithm is not expected to find a common factor if two roots of the same polynomial are very close.

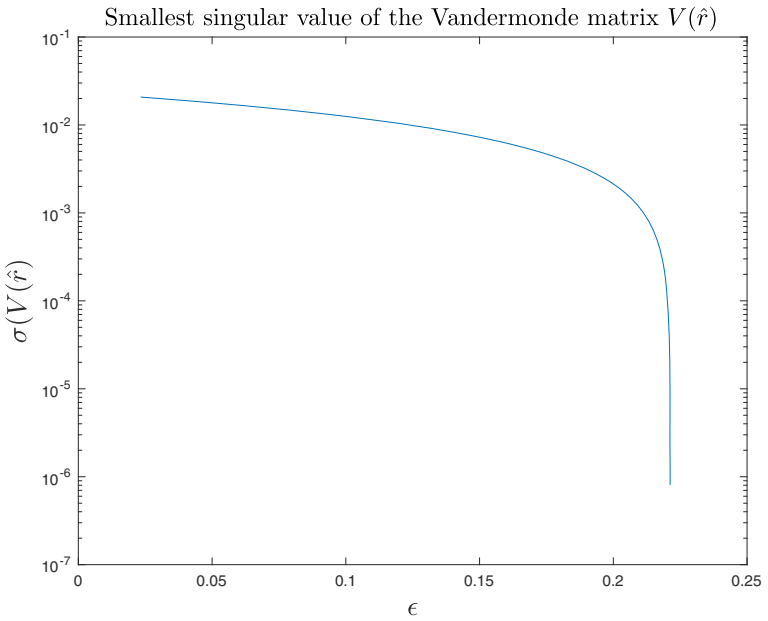

Fig. 1 Smallest singular value of the Vandermonde matrix $V(\tilde{r})$ as function of ϵ

computation problem, but the associated optimization problem is based on the distance between the polynomials coefficients. The data are still the polynomials in (13) and the numerical results are compared in Table 1:

As a further evidence, we observe how the numerical results reflect the two different formulations of the problem: in the comparison, the Sylvester low-rank approximation keeps the distance between the coefficients smaller, while the distance between the computed roots is smaller for the Vandermonde formulation of the problem. This does not mean that one algorithm is better than the other to approach the problem, but it is up to the reader to choose the formulation which best suits the considered application. What motivated the new problem formulation is the property that a roots-based distance is independent of the polynomials coefficients (the latter are not unique!).

Table 1 Numerical comparison for an approximate common factor computation of degree 1 between the polynomials in (13)

	$\|a - \tilde{a}\|_2$	$\|r - \tilde{r}\|_2$	$(\tilde{r}_1, \tilde{r}_2)$
Sylvester low-rank appr. [13]	0.051	0.345	(0.952, 2.401, 2.401, 4.259)
Vandermonde low-rank appr.	0.746	0.221	(1.060, 2.353, 2.353, 3.982)

Comparison between two algorithms which solve two different formulations of the problem. From left to right, the columns show: distance between coefficients, distance between roots and computed roots

4.2 Identification of time series generated by nonlinear maps

We show here a numerical simulation for the problem described in Section 2.2. The exact (noiseless) data is a vector of 20 points generated by the following function:

$$X_{n+1} = 1 - 1.4X_n^2 + 0.3X_{n-1} - 0.5e^{X_n} \quad n = 1, \ldots, 20, \tag{17}$$

and starting from the point $(X_0, X_1) = (0, 0)$. First of all, we need to build an appropriate structured matrix, as in (8). We observe that on the right hand side of (17), there are a polynomial of degree 2 and an exponential. Following Section 2.2.2, the structured matrix is built as follows: the linear block is a Hankel matrix with 3 columns associated with the variables X_{n+1}, X_n, X_{n-1}, while the (nonlinear) functions f_i are all the monomials in the variables X_n, X_{n-1} of degree 2 and an exponential function (we assume to know this partial information about the functions generating the time series).

$$\mathcal{S}(X) = (\mathbf{1}, X_{n+1}, X_n, X_{n-1}, X_{n-1}^2, X_n^2, X_{n-1}X_n, e^{X_n}). \tag{18}$$

We observe that a block of the matrix is Hankel (it is associated with the linear part of the system) while the other columns correspond to nonlinear functions. If we plug the time series (17) into the matrix (18), we get a rank-deficient structured matrix. Similarly to the previous section, we can compute the structures of the matrices E and $\dot{E}$ (needed in Algorithm 1) starting from $S(X)$: the latter has the form

$$\begin{aligned}\dot{E}(X, e, \dot{e}) =& (0, \dot{e}_{n+1}, \dot{e}_n, \dot{e}_{n-1}, 2(X_{n-1} + e_{n-1})\dot{e}_{n-1}, 2(X_n + e_n)\dot{e}_n, \\ & (X_{n-1} + e_{n-1})\dot{e}_n + (X_n + e_n)\dot{e}_{n-1}).\end{aligned}$$

The goal of this experiment is to perturb the starting time series with noisy terms of increasing magnitude, and then try to recover it by solving a nonlinear structured low-rank approximation problem using the proposed algorithm; doing so, we observe how the error on the computed solutions changes with the magnitude of the perturbations.

The perturbation of each entry of the vector is built by adding random noisy terms as follows:

$$W = X + \alpha * \frac{r}{\|r\|_2} * \|X\|_2, \tag{19}$$

where r is a random vector of the same size of X whose entries come from i.i.d. standard normal distributions and α is a scalar which changes the magnitude of the noise.

In Fig. 2, we uniformly increase the value of α in the interval $[0, 0.5]$ to observe how the relative error $\|\tilde{X} - X\|_2/\|X\|_2$ changes ($\tilde{X}$ is the computed solution).

We can observe the linear growth of the relative error with the increasing impact of the noise on the exact data.

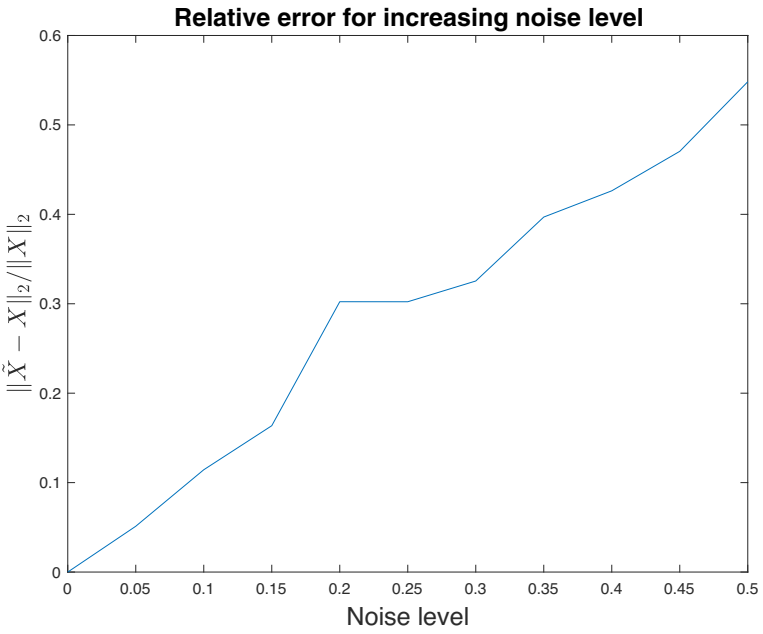

Fig. 2 Identification of a time series generated by a nonlinear function: relative error for increasing noise level averaged over 10 different perturbations

5 Conclusion

Motivated by two important applications in computer algebra and system identification, we proposed an algorithm to solve structured low-rank approximation problems in the case the structure of the involved matrix is nonlinear. This first version of the proposed optimization algorithm comes from the extension of a method presented in the literature for linearly structured matrices, and it is based on the (structured) minimization of the smallest singular value of the involved matrix. The motivating applications are the following.

1. The computation of approximate polynomials common factors solving a Vandermonde low-rank approximation problem: the use of the Vandermonde matrix (instead of the classical Sylvester matrix) allows to measure the distance between the polynomials roots (instead of the coefficients). The roots of a polynomial are invariant under the multiplication by a constant.
2. Identification of a system generated by a nonlinear map: a well-known result in system theory states that the order of a linear time-invariant system equals the rank of a Hankel matrix built from an observed trajectory. If the system is nonlinear, we need to replace the Hankel matrix with a structured matrix whose columns correspond to all the nonlinear functions generating system trajectory. The rank deficiency of such a structured matrix detects the dependence between the linear and the nonlinear elements.

Possible future work can focus on the replacement of Algorithm 1 with a different method which guarantees the structure preserving property as well as on the general improvement of the computational performance of the whole algorithm.

Acknowledgements The author is thankful to an anonymous referee for several constructive comments which improved the quality of the paper.

Funding The research leading to these results has received funding from the Fond for Scientific Research Vlaanderen (FWO) projects G028015N and G090117N and the FNRS-FWO under Excellence of Science (EOS) Project no 30468160 "Structured low-rank matrix/tensor approximation: numerical optimization-based algorithms and applications."

Data availability All data generated or analyzed during this study are included in this published article.

Declarations

Conflict of interest The author declares no competing interests.

References

1. Eckart, C., Young, G.: The approximation of one matrix by another one of lower rank. Psychometrika **1**, 211–218 (1936). https://doi.org/10.1007/BF02288367
2. Markovsky, I. Low-rank Approximation: Algorithms, Implementation, Applications, 2nd edn. Springer, Berlin (2019)
3. Chu, M.T., Funderlic, R.E., Plemmons, L.J.: Structured low rank approximation. Linear Algebra Its Appl. **366**, 157–172 (2003). https://doi.org/10.1016/S0024-3795(02)00505-0
4. De Moor, B.: Total least squares for affinely structured matrices and the noisy realization problem. IEEE Trans. Signal Process. **42**, 3104–3113 (1994). https://doi.org/10.1109/78.330370
5. Usevich, K., Markovsky, I.: Variable projection for affinely structured low-rank approximation in weighted 2-norms. J. Comput. Appl. Math. **272**, 430–448 (2014). https://doi.org/10.1016/j.cam.2013.04.034
6. Shklyar, S., Kukush, A., Markovsky, I., Van Huffel, S.: On the conic section fitting problem. J. Multivar. Anal. **98**, 588–624 (2007). https://doi.org/10.1016/j.jmva.2005.12.003
7. Markovsky, I., Kukush, A., Van Huffel, S.: Consistent least squares fitting of ellipsoids. Numer. Math. **98**, 177–194 (2004). https://doi.org/10.1007/s00211-004-0526-9
8. Markovsky, I., Usevich, K.: Nonlinearly structured low-rank approximation. In: Fu, Y.R. (ed.) Low-Rank and Sparse Modeling for Visual Analysis, pp. 1–22. Springer (2014). https://doi.org/10.1007/978-3-319-12000-3_1
9. Usevich, K., Markovsky, I.: Adjusted least squares fitting of algebraic hypersurfaces. Linear Algebra Appl. **502**, 243–274 (2014). https://doi.org/10.1016/j.laa.2015.07.023
10. Rosen, J.B., Park, H., Glick, J.: Structured total least norm for nonlinear problems. SIAM J. Matrix Anal. Appl. **20**(1), 14–30 (1998). https://doi.org/10.1137/S0895479896301662
11. Lemmerling, P., Van Huffel, S., De Moor, B.: The structured total least-squares approach for nonlinearly structured matrices. Numer. Linear Algebra Appl. **9**, 321–332 (2002). https://doi.org/10.1002/nla.276
12. Fazzi, A., Kukush, A., Markovsky, I.: Bias correction for Vandermonde low-rank approximation. Econ. Stat. In press. https://doi.org/10.1016/j.ecosta.2021.09.001 (2021)
13. Fazzi, A., Guglielmi, N., Markovsky, I.: An ODE based method for computing the approximate greatest common divisor of polynomials. Numer. Algorithms **81**, 719–740 (2019). https://doi.org/10.1007/s11075-018-0569-0
14. Guglielmi, N., Markovsky, I.: An ODE based method for computing the distance of coprime polynomials to common divisibility. SIAM J. Numer. Anal. **55**, 1456–1482 (2017). https://doi.org/10.1137/15M1018265

15. Fazzi, A., Guglielmi, N., Markovsky, I.: Generalized algorithms for the approximate matrix polynomial GCD of reducing data uncertainties with application to MIMO system and control. J. Comput. Appl. Math. 393. https://doi.org/10.1016/j.cam.2021.113499 (2021)
16. Fazzi, A., Guglielmi, N., Markovsky, I.: A gradient system approach for Hankel structured low-rank approximation. Linear Algebra its appl. **623**, 236–257 (2021). https://doi.org/10.1016/j.laa.2020.11.016
17. Barkhuijsen, H., De Beer, R., Van Ormondt, D.: Improved algorithm for noniterative time-domain model fitting to exponentially damped magnetic resonance signals. J. Magentic Resonance **73**, 553–557 (1987). https://doi.org/10.1016/0022-2364(87)90023-0
18. Markovsky, I., Fazzi, A., Guglielmi, N.: Applications of polynomial common factor computation in signal processing. In: et al, Y.D. (ed.) Latent Variable Analysis and Signal Separation. Lecture Notes in Computer Science, vol. 10891, pp. 99–106. Springer (2018). https://doi.org/10.1007/978-3-319-93764-9_10
19. Fazzi, A., Guglielmi, N., Markovsky, I.: Computing common factors of matrix polynomials with applications in system and control theory. In: Proc. of the IEEE Conf. on Decision and Control, Nice, France, pp. 7721–7726 (2019). https://doi.org/10.1109/CDC40024.2019.9030137
20. De Iuliis, V., Germani, A., Manes, C.: Identification of forward and feedback transfer functions in closed-loop systems with feedback delay. IFAC-PapersOnLine **50**(1), 12847–12852 (2017). https://doi.org/10.1016/j.ifacol.2017.08.1935
21. Markovsky, I., Van Huffel, S.: An algorithm for approximate common divisor computation. In: Proceedings of MTNS'06, Kyoto, Japan, pp. 274–279 (2006)
22. Pan, V.Y.: Computation of approximate polynomial GCDs and an extension. Inf. Comput. **167**, 71–85 (2001). https://doi.org/10.1006/inco.2001.3032
23. Qiu, W., Hua, Y., Abed-Meraim, K.: A subspace method for the computation of the GCD of polynomials. Automatica **33**, 741–743 (1997). https://doi.org/10.1016/S0005-1098(96)00244-0
24. Terui, A.: GPGCD: An iterative method for calculating approximate GCD of univariate polynomials. Theoret. Comput. Sci. **479**, 127–149 (2013). https://doi.org/10.1007/978-3-642-15274-0_22
25. Usevich, K., Markovsky, I.: Variable projection methods for approximate (greatest) common divisor computations. Theor. Comput. Sci. **681**, 176–198 (2017). https://doi.org/10.1016/j.tcs.2017.03.028
26. Zeng, Z., Dayton, B.H.: The approximate GCD of inexact polynomials, Part I: a univariate algorithm. In: Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation, pp. 320–327. ACM (2004). https://doi.org/10.1145/1005285.1005331
27. Markovsky, I., Willems, J.C., Van Huffel, S., De Moor, B.: Exact and approximate modeling of linear systems: a behavioral approach. SIAM (2006)
28. Berberich, J., Allgöwer, F.: A trajectory-based framework for data-driven system analysis and control. In: Eur. Control Conf, pp. 1365–1370 (2020). https://doi.org/10.23919/ECC51009.2020.9143608
29. Rueda-Escobedo, J.G., Schiffer, J.: Data-driven internal model control of second-order discrete volterra systems. In: IEEE Conf. Decision Control, pp. 4572–4579 (2020). https://doi.org/10.1109/CDC42340.2020.9304122
30. Mishra, V., Markovsky, I., Fazzi, A., Dreesen, P.: Data-driven simulation for NARX Systems. In: 2021 29Th European Signal Processing Conference (EUSIPCO), pp. 1055–1059 (2021). https://doi.org/10.23919/EUSIPCO54536.2021.9616226
31. Vaidya, P.G., Anand, P.S.S., Nagaraj, N.: A nonlinear generalization of singular value decomposition and its applications to mathematical modeling and chaotic cryptanalysis. Acta Appl. Math. **112**, 205–221 (2010). https://doi.org/10.1007/s10440-010-9560-z
32. Golub, G.H., Van Loan, C.F. Matrix Computations, 4th edn. The Johns Hopkins University Press, Baltimore (2013)
33. Andrew, A.L., Chu, K.-W.E., Lancaster, P.: Derivatives of eigenvalues and eigenvectors of matrix functions. SIAM J. Matrix Anal. Appl. **14**(4), 903–926 (1993). https://doi.org/10.1137/0614061